PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-282695
(43)Date of publication of application : 15.10.1999

(51)Int.Cl. G06F 9/46
G06F 9/46

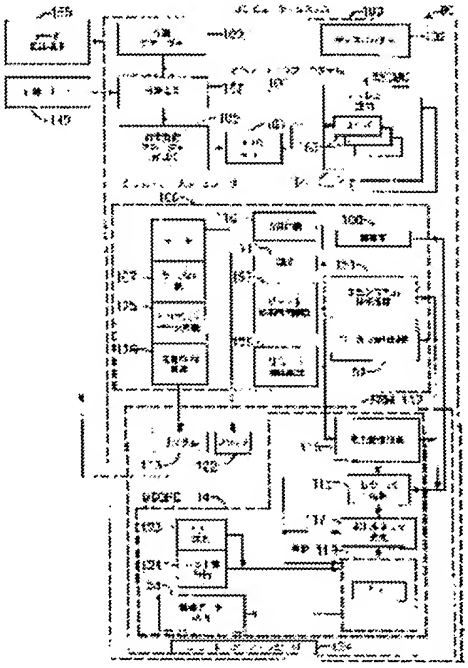
(21)Application number : 11-035205
(22)Date of filing : 15.02.1999

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : YOCUM PETER B
EILERT CATHERINE K
ARWE JOHN E

(30)Priority
Priority number : 98 38573 Priority date : 18.03.1998 Priority country : US

(54) METHOD AND DEVICE FOR CONTROLLING NUMBER OF SERVERS IN MULTISYSTEM CLUSTER

(57)Abstract:
PROBLEM TO BE SOLVED: To control the number of servers in a multisystem cluster.
SOLUTION: A computer system 100 has a queue 161 for organizing an inputted work request as a service class, transferring respective service classes to a cluster and serving them by servers 163. Prescribed performance indexes are previously allocated to respective service classes. Each system 100 selects a certain service class as a donor class for providing a system resource based on a degree of coincidence of each service class with its own target and selects another service class as a receiver class for receiving the system resource. When a resource bottleneck causing the receiver class to lose its target is the number of servers, each system 100 determines whether several servers are to be added to the receiver class or not based on judgement whether a plus effect of server addition to the performance index of the receiver class exceeds the minus effect applied to the performance index of a donor class or not.



LEGAL STATUS

[Date of request for examination] 09.08.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]	3121584
[Date of registration]	20.10.2000
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平11－282695

(43)公開日　平成11年(1999)10月15日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/46	３４０	Ｇ０６Ｆ　9/46	３４０Ｄ
	３６０		３６０Ｃ

審査請求　未請求　請求項の数17　ＯＬ　（全 20 頁）

(21)出願番号　特願平11－35205

(22)出願日　平成11年(1999)２月15日

(31)優先権主張番号　０９／０３８５７３

(32)優先日　1998年３月18日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　ピーター・ビィ・ヨクム
アメリカ合衆国12590、ニューヨーク州ワッピンガーズ・フォールズ、ワイルドウッド・マナー　17ビィ

(74)代理人　弁理士　坂口　博　（外１名）

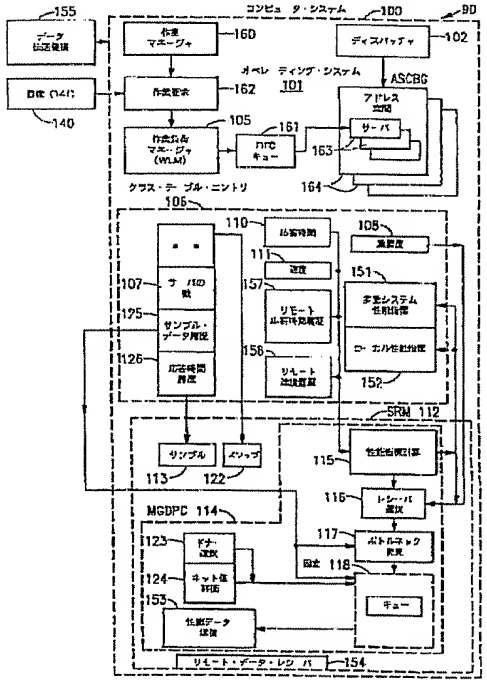
最終頁に続く

(54)【発明の名称】　多重システム・クラスタ内のサーバの数を制御する方法及び装置

(57)【要約】

【課題】　多重システム・クラスタ内のサーバの数を制御する方法及び装置の提供。

【解決手段】　入来作業要求がサービス・クラスに編成され、それらの各々がクラスタに渡って、サーバによりサービスされるキューを有する。各サービス・クラスは予め、所定の性能指標を割当てられる。各システムはサービス・クラスがそれらの目標にどれ程良く合致するかにもとづき、あるサービス・クラスを、システム資源を提供するドナー・クラスとして選択し、別のサービス・クラスを、システム資源を受け取るレシーバ・クラスとして選択する。各システムは次に、レシーバ・クラスがその目標を逸する原因となる資源ボトルネックがサーバの数の場合、各システムは、サーバの追加がレシーバ・クラスの性能指標に及ぼすプラスの効果が、ドナー・クラスの性能指標に及ぼすマイナスの効果を上回るか否かにもとづき、幾つのサーバがレシーバ・クラスに追加されるべきかを決定する。



【特許請求の範囲】
【請求項１】情報処理システムのクラスタ内においてサーバの数を制御する方法であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の１つ以上のサーバにより処理されるものにおいて、
１つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定するステップと、
１つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップと、
前記サーバを前記ターゲット・システム上に追加するステップとを含む、方法。
【請求項２】前記決定するステップが前記システムの各々により実行される、請求項１記載の方法。
【請求項３】前記システムの各々が、該システムがターゲット・システムか否かを決定し、前記ターゲット・システムであると決定するとき、１つ以上のサーバを局所的に追加する、請求項２記載の方法。
【請求項４】前記サービス・クラスが第１のサービス・クラスであり、前記クラスタが少なくとも１つの他のサービス・クラスを有し、前記サービス・クラスの各々が、それぞれに対して定義された性能測定を有する、請求項１記載の方法。
【請求項５】サーバが前記第１のサービス・クラスに追加されるべきか否かを決定するステップが、
所定数のサーバを前記第１のサービス・クラスに追加することが、前記第１のサービス・クラスの前記性能測定に及ぼすプラスの効果を決定するステップと、
前記所定数のサーバを前記第１のサービス・クラスに追加することが、１つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を決定するステップと、
前記第１のサービス・クラスの前記性能測定に及ぼすプラスの効果が、１つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を上回るか否かを決定するステップとを含む、請求項４記載の方法。
【請求項６】サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、
前記クラスタ内の任意のシステムが、１つ以上の追加のサーバを追加するための十分な遊休容量を有するか否かを決定するステップと、
前記クラスタ内の任意のシステムが、１つ以上の追加のサーバを追加するための十分な遊休容量を有する場合、１つのこうしたシステムを前記ターゲット・システムとして選択するステップとを含む、請求項１記載の方法。
【請求項７】最大の未使用容量を有するシステムが前記ターゲット・システムとして選択される、請求項６記載の方法。
【請求項８】サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、
前記クラスタ内のいずれのシステムも、１つ以上の追加のサーバを追加するための十分な遊休容量を有さない場合、１つ以上のサーバの追加が他の作業に最小の影響を及ぼす前記クラスタ内のシステムを決定するステップと、
そのシステムを前記ターゲット・システムとして選択するステップとを含む、請求項６記載の方法。
【請求項９】マシンにより読出され、前記マシンにより実行されて、請求項１記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。
【請求項１０】情報処理システムのクラスタ内においてサーバの数を制御する装置であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の１つ以上のサーバにより処理されるものにおいて、
１つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定する手段と、
１つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定する手段と、
前記サーバを前記ターゲット・システム上に追加する手段とを含む、装置。
【請求項１１】情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する方法であって、入来作業要求が前記キューに配置され、前記システム上の１つ以上のサーバにより処理されるものにおいて、
前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップと、
前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動するステップとを含む、方法。
【請求項１２】前記決定するステップが周期的インタバルで実行される、請求項１１記載の方法。
【請求項１３】異なるサービス・クラスに割当てられる作業要求が、異なるキューに配置され、前記決定するステップが前記キューの各々に対して実行される、請求項１１記載の方法。
【請求項１４】前記決定するステップが、前記クラスタ内の各システムにより実行される、請求項１１記載の方法。
【請求項１５】前記決定するステップが、
前記キューが前記システム上にサーバを有するか否かを決定するステップと、

前記キューが前記システム上にサーバを有さない場合、前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップとを含む、請求項１４記載の方法。

【請求項１６】マシンにより読出され、前記マシンにより実行されて、請求項１１記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。

【請求項１７】情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する装置であって、入来作業要求が前記キューに配置され、前記システム上の１つ以上のサーバにより処理されるものにおいて、

前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定する手段と、

前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動する手段とを含む、装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、第１のサービス・クラスに属する入来作業要求が、１つ以上のサーバによる処理のために、キューに配置される情報処理システムにおいて、サーバの数を制御する方法及び装置に関する。

【０００２】

【従来の技術】使用可能なサーバへの割当てのために、入来作業要求がキューに配置されるシステムは周知である。入来作業要求が到来する頻度は、容易に制御され得ないので、こうしたキュー待機型システムにおいて、システム性能（キュー遅延などにより測定される）を制御する基本手段は、サーバの数を制御することである。従って、サービスされるキューの長さが特定の高いしきい値に達するとき、追加のサーバを始動したり、サービスされるキューの長さが特定の低いしきい値に達するとき、サーバを停止することが知られている。こうした手段は、その設計目的を達成するが、キューに待機された作業要求に加え、他の作業単位がシステム資源を競合するシステムでは不十分である。従って、たとえキューに対して追加のサーバを提供することで、そのキュー内における作業要求の性能を向上しても、こうしたサーバの提供は、システムにより処理される他の作業単位の性能を低下し得、システムの性能を全体として悪化し得る。

【０００３】ほとんどの今日のオペレーティング・システム・ソフトウェアは、作業要求に対して指定されるエンドユーザ指向の目的に従い、サーバの数を管理し、同一のコンピュータ・システム内で実行される、独立の目標を有する他の作業を考慮する責任を負うことができない。

【０００４】本願の出願人に権利譲渡された１９９７年３月２８日付けのJ. D. Amanらによる継続中の米国特許出願第８２８４４０号は、特定のシステムにおいて、サーバの数を制御する方法及び装置を開示し、そこでは第１のサービス・クラスに属する入来作業要求が、１つ以上のサーバによる処理のためにキューに配置される。本システムは更に、システム資源のドナー（donor）として作用する、少なくとも１つの他のサービス・クラスに割当てられる作業単位を有する。本発明によれば、性能測定が第１のサービス・クラスの他に、少なくとも１つの他のサービス・クラスに対して定義される。サーバを第１のサービス・クラスに追加する以前に、第１のサービス・クラスの性能測定に及ぼすプラスの効果だけでなく、他のサービス・クラスの性能測定に及ぼすマイナスの効果も決定される。第１のサービス・クラスの性能測定に及ぼすプラスの効果が、他のサービス・クラスの性能測定に及ぼすマイナスの効果に勝る場合に限り、サーバが第１のサービス・クラスに追加される。

【０００５】

【発明が解決しようとする課題】この継続中の特許出願で開示される発明は、サーバを追加するか否かを決定するとき、他の作業に対する影響を考慮するが、これは単一システムの状況においてそのようにする。しかしながら、多重システム・コンプレックス（"シスプレックス（sysplex）"）では、作業要求の１つのキューが、コンプレックスに渡り複数のサーバによりサービスされる。従って、所与のキューに対して、サーバを追加するか否かだけでなく、シスプレックス全体の性能を最適化するために、サーバをどこに追加すべきかが決定され得る。

【０００６】

【課題を解決するための手段】本発明は、情報処理システムのクラスタ内のサーバの数を制御する方法及び装置に関して、そこでは第１のサービス・クラスに属する入来作業要求が、１つ以上のサーバによる処理のためにキューに配置される。入来作業要求のあるものは、クラスタ内のサーバのサブセット上でのみ実行されるための要求を有し得る。こうした要求を有する作業要求は、それらが実行されなければならないクラスタ内のシステムのサブセットに対して、"親和性（affinity）"を有すると言われる。本発明によれば、サーバがクラスタ内の１つ以上のシステム上で始動され、キュー内の作業要求を処理する。これらのサーバを始動するシステムは、システムのクラスタの全容量を利用するように、及び作業要求の親和性要求に応じるように、更にクラスタ内のシステム上で実行されているかも知れない他の作業への影響を最小化するように選択される。新たなサーバが始動されるシステムはまた、システム資源のドナーとして作用する第２のサービス・クラスに割当てられる作業単位を有する。本発明によれば、性能測定が第１のサービス・ク

ラス同様、第２のサービス・クラスに対しても定義される。サーバを第１のサービス・クラスに追加する以前に、第１のサービス・クラスの性能測定に及ぼすプラスの効果だけでなく、第２のサービス・クラスの性能測定に及ぼすマイナスの効果も決定される。第１のサービス・クラスの性能測定に及ぼすプラスの効果が、第２のサービス・クラスの性能測定に及ぼすマイナスの効果に勝る場合に限り、サーバが第１のサービス・クラスに追加される。

【０００７】本発明は、複数のユーザ性能目標クラスの各々に対して、各目標クラスの性能目標にもとづき、システムのクラスタに渡るサーバの数のシステム管理を可能にする。競合する目標クラスに対して、サーバの追加または除去の影響を考慮するトレードオフが提供される。

【０００８】

【発明の実施の形態】本発明を組み込むシステムについて述べる前準備として、作業負荷管理（本発明がそれ上で構成される）の概念に関して前置きすることが適切であろう。

【０００９】作業負荷管理は、オペレーティング・システムにより管理される作業単位（プロセス、スレッドなど）が、クラス（サービス・クラスまたは目標クラスと呼ばれる）に編成される概念であり、クラスは予め定義された目標にどれ程よく合致するかに従い、システム資源を提供される。ドナー・クラスからレシーバ（receiver）・クラスへの資源の再割当ては、こうした再割当てから得られるレシーバ・クラスの性能の改善が、ドナー・クラスの性能の劣化に勝る場合、すなわち、予め定義された性能基準により決定される性能における、正味のプラス効果が存在する場合に実行される。このタイプの作業負荷管理は、資源の割当てが資源が再割当てされる作業単位への効果だけにより決定されるのではなく、資源が奪われる作業単位への効果によっても決定されるという点で、大部分のオペレーティング・システムにより実行される"普通の（run-of-the-mill）"資源管理とは異なる。

【００１０】この一般的なタイプの作業負荷マネージャは、以下に挙げる特許、特許出願、及び特許以外の刊行物において開示されている。

【００１１】D. F. Fergusonらによる米国特許出願第５５０４８９４号"Workload Managerfor Achieving Transaction Class Response Time Goals in a MultiprocessingSystem"。

J. D. Amanらによる米国特許第５４７３７７３号"Apparatus and Method forManaging a Data Processing System Workload According to Two or More Distinct Processing Goals"。

C. K. Eilertらによる米国特許第５５３７５４２号"Apparatus and Method for Managing a Server Workload According to Client Performance Goals in aClient/Server Data Processing System"。

J. D. Amanらによる米国特許第５６０３０２９号"System of Assigning WorkRequests Based on Classifying into an Eligible Class Where the Criterials Goal Oriented and Capacity Information is Available"。

C. K. Eilertらによる米国特許第５６７５７３９号"Apparatus and Method for Managing a Distributed Data Processing System Workload According to aPlurality of Distinct Processing Goal Types"。

１９９５年２月３日付けのC. K. Eilertらによる米国特許出願第３８３０４２号"Multi-System Resource Capping"。

１９９５年６月７日付けのJ. D. Amanらによる米国特許出願第４８８３７４号"Apparatus and Accompanying Method for Assigning Session Requests in a Multi-Server Sysplex Environment"。

１９９７年３月２８日付けのJ. D. Amanらによる米国特許出願第８２８４４０号"Method and Apparatus for Controlling the Number of Servers in a Client/Server System"。

MVS Planning：Workload Management、ＩＢＭ刊行物GC28-1761-00、１９９６年。

MVS Programming：Workload Management Services、ＩＢＭ刊行物GC28-1773-00、１９９６年。

【００１２】前記特許及び刊行物の中で、米国特許第５５０４８９４号及び同第５４７３７７３号は、基本的な作業負荷管理システムを開示し、米国特許第５５３７５４２号は、米国特許第５４７３７７３号の作業負荷管理システムの、クライアント／サーバ・システムへの特定のアプリケーションを開示し、米国特許第５６７５７３９号及び米国特許出願第３８３０４２号は、米国特許第５４７３７７３号の作業負荷管理システムの、複数の相互接続システムへの特定のアプリケーションを開示し、米国特許第５６０３０２９号は、多重システム・コンプレックス（"シスプレックス"）における作業要求の割当てに関して、米国特許出願第４８８３７４号は、こうしたコンプレックスにおけるセッション要求の割当てに関して、前述のように、米国特許出願第８２８４４０号は、多重システム・コンプレックスの１つのシステム上でのサーバの数の制御に関する。特許以外の２つの刊行物は、ＩＢＭ（登録商標）ＯＳ／３９０（商標）（以前はＭＶＳ（登録商標））オペレーティング・システムにおける作業負荷管理の実現について述べている。

【００１３】図１は、本発明の一般的な実施例における主要な環境及びフィーチャを示し、相互接続され、協働するコンピュータ・システム１００のクラスタ９０を含み、それらの一般的な２つが示されている。本発明の環境は、作業要求１６２のキュー１６１と、クラスタ９０に渡り分散され、作業要求をサービスするサーバ１６３

のプールとを含む。本発明は、キューに待機された作業の性能目標クラスと、コンピュータ・システム１００内で競合する作業の性能目標クラスとにもとづき、サーバ１６３の数の管理を可能にする。コンピュータ・システム１００のクラスタ９０に対して１つのポリシを有することは、分散された作業負荷の単一イメージ・ビューを提供するのに役立つ。当業者であれば、任意の数のコンピュータ・システム１００、及び１つのコンピュータ・システム１００内の任意の数のこうしたキュー１６１及びサーバ１６３のグループが、本発明の趣旨及び範囲から逸れることなく使用され得ることが理解されよう。

【００１４】コンピュータ・システム１００は分散された作業負荷を実行し、各コンピュータ・システムが、例えばＩＢＭ　ＯＳ／３９０オペレーティング・システムなどの、オペレーティング・システム１０１の自身のコピーにより制御される。

【００１５】それぞれのコンピュータ・システム１００上のオペレーティング・システム１０１の各コピーは、本願で述べられるステップを実行する。説明の中で"ローカル"・システム１００を指し示すとき、それは述べられているステップを実行しているシステム１００を意味する。一方、"リモート"・システム１００は、管理される他の全てのシステム１００を意味する。各システム１００がそれ自身を局所的（ローカル）と見なし、他の全てのシステム１００を遠隔的と見なす点に留意されたい。

【００１６】本発明に関する改良以外は、システム１００は、継続中の米国特許出願第８２８４４０号及び米国特許第５６７５７３９号で開示されるシステムと類似である。図１に示されるように、システム１００は複数の相互接続されるシステム１００の１つであり、これらは同様に管理され、クラスタ９０（システム・コンプレックスまたはシスプレックスとも呼ばれる）を構成する。米国特許第５６７５７３９号で教示されるように、作業単位が分類される様々なサービス・クラスの性能が、特定のシステム１００に対してだけでなく、クラスタ９０全体に対しても追跡され得る。この目的のために、また後述の説明から明らかになるように、システム１００とクラスタ９０内の他のシステム１００との間で、性能結果を伝達し合う手段が提供される。

【００１７】ディスパッチャ１０２は、オペレーティング・システム１０１の構成要素であり、次にコンピュータにより実行される作業単位を選択する。作業単位は、コンピュータ・システム１００の目的に相当する有用な作業を実行するアプリケーション・プログラムである。実行準備が整った作業単位は、アドレス空間制御ブロック（ＡＳＣＢ）・キューと呼ばれる、オペレーティング・システム・メモリ内の制御ブロックの連鎖により表される。

【００１８】作業マネージャ１６０は、オペレーティング・システム１０１の外部の構成要素であり、これはオペレーティング・システム・サービスを用いて、１つ以上のキュー１６１を作業負荷マネージャ（ＷＬＭ）１０５に定義し、作業要求１６２をこれらのキュー上に挿入する。作業マネージャ１６０は、挿入された作業要求１６２を、クラスタ９０内の任意のシステム１００上の作業マネージャ１６０のサーバ１６３による選択のために、先入れ先出し（ＦＩＦＯ）順に保持する。作業マネージャ１６０は、サーバが、それが実行されているシステム１００と親和性を有する要求だけを選択することを保証する。

【００１９】サーバ１６３は作業マネージャ１６０の構成要素であり、キューに待機された作業要求１６２をサービスすることができる。作業負荷マネージャ１０５がサーバ１６３を始動し、作業マネージャ１６０のキュー１６１上の作業要求１６２をサービスするとき、作業負荷マネージャ１０５は、共用データ機構１４０上に記憶されたサーバ定義を用いて、アドレス空間（すなわちプロセス）１６４を始動する。作業負荷マネージャ１０５により始動されるアドレス空間１６４は、作業負荷マネージャ１０５の指示に従い、そのアドレス空間がサービスすべき特定のキュー１６１上の要求１６２をサービスする、１つ以上のサーバ（すなわちディスパッチ可能単位またはタスク）１６３を含む。

【００２０】図２は、コンピュータ・システム１００が接続されるネットワーク（図示せず）から、作業負荷マネージャ１０５により管理されるサーバ・アドレス空間１６４への、クライアント作業要求１６２の流れを示す。作業要求１６２はクラスタ９０内の特定のコンピュータ・システム１００に経路指定され、作業マネージャ１６０により受信される。作業要求１６２の受信に際して、作業マネージャ１６０はそれを作業負荷マネージャ（ＷＬＭ）・サービス・クラスに分類し、作業要求を作業キュー１６１に挿入する。作業キュー１６１は、クラスタ９０内の全てのコンピュータ・システム１００により共用される。すなわち、作業キュー１６１はクラスタ全体に渡るキューである。作業要求１６２は、それを実行する準備が整ったサーバが現れるまで、作業キュー１６１内で待機する。

【００２１】新たな作業要求１６２を実行する準備が整った（アドレス空間が丁度始動されたか、タスクが前の要求の実行を終了したとき）、クラスタ９０内のあるシステム１００上のサーバ・アドレス空間１６４内のタスク１６３が、新たな作業要求のために作業マネージャ１６０を呼び出す。アドレス空間１６４がサービスしている作業キュー１６１上に作業要求１６２が存在し、その作業要求がサーバが実行されているシステム１００と親和性を有する場合、作業マネージャ１６０はその作業要求をサーバ１６３に受け渡す。それ以外では、作業マネージャ１６０は作業要求１６２が有効になるまで、サー

バ１６３を延期する。

【００２２】作業要求１６２が作業マネージャ１６０により受信されるとき、それは作業キュー１６１上に配置され、サーバ１６３が作業要求を実行するために使用可能になるのを待機する。作業要求１６２の作業マネージャ１６０、アプリケーション環境名、及びＷＬＭサービス・クラスのそれぞれの固有の組み合わせに対して、１つの作業キュー１６１が存在する。（アプリケーション環境は、類似のクライアント作業要求１６２のセットが実行されるべき環境である。ＯＳ／３９０条件では、これは作業要求を実行するために、サーバ・アドレス空間を始動するために使用されるジョブ制御言語（ＪＣＬ）プロシージャにマップされる。）特定の作業キュー１６１に対して、第１の作業要求１６２が到来するとき、キューイング構造が動的に生成される。作業キュー１６１に対して、所定時間（例えば１時間）何も活動が発生しなかった場合、構造は消去される。新たなＷＬＭポリシを活動化するなどのように、キューに待機された作業要求１６２のＷＬＭサービス・クラスを変更し得るアクションが発生する場合、作業負荷マネージャ１０５は作業マネージャ１６０にその変更を知らせ、作業マネージャ１６０は、各作業要求１６２の新たなＷＬＭサービス・クラスを反映するように、作業キュー１６１を再生する。

【００２３】サーバ１６３を有さないシステム１００と親和性を有する作業要求１６２は、他のシステム１００上に、その作業要求のサービス・クラスの目標を満たす上で十分なサーバが存在する場合、決して実行されないといった危険性が存在する。この危険性を回避するために、作業負荷マネージャ１０５は、クラスタ９０内のどこかに、キュー１６１上の各作業要求を実行可能な少なくとも１つのサーバ１６３が存在するように保証する。図８はこの論理を示す。この論理は、クラスタ９０内の各コンピュータ・システム１００上の作業マネージャ１６０により実行される。

【００２４】ステップ７０１で、作業マネージャ１６０が自身が所有する第１のキュー１６１を調査する。ステップ７０２で、作業マネージャ１６０が、ローカル・システム１００上にこのキュー１６１に対するサーバ１６３が存在するか否かをチェックする。サーバ１６３が存在する場合、作業マネージャ１６０は次のキュー１６１に移行する（ステップ７０８乃至７０９）。作業マネージャ１６０が、局所的にサーバ１６３を有さないキュー１６１を見い出す場合、作業マネージャ１６０は次にキュー上の各作業要求を調査し、第１の作業要求を開始する（ステップ７０３乃至７０６）。各作業要求１６２に対して、作業マネージャ１６０は、クラスタ９０内のどこかに、現作業要求を実行可能なサーバ１６３が存在するか否かをチェックする（ステップ７０４）。現作業要求を実行可能なサーバ１６３が存在する場合、作業マネージャ１６０はキュー１６１上の次の作業要求１６２に移行する（ステップ７０６）。現作業要求を実行可能なサーバ１６３が存在しない場合、作業マネージャ１６０は作業負荷マネージャ１０５を呼び出し、サーバ１６３を始動し（ステップ７０７）、次のキュー１６１に移行する（ステップ７０８乃至７０９）。作業マネージャ１６０は、作業マネージャにより所有される全てのキュー１６１が処理されるまで（ステップ７１０）、同様に継続する。

【００２５】作業マネージャ１６０が作業負荷マネージャ１０５を呼び出すとき（ステップ７０７）、作業要求に対してサーバを始動すべき最善のシステム１００を決定するために、作業負荷マネージャ１０５は、クラスタ９０内の各システム１００に対して、各重要度にて有効なサービス、及びそのシステムに対して未使用のサービスを示す、サービス有効配列（Service Available Array）を保持する。この配列は下記のように、各重要度に対するエントリ、及び未使用のサービスに対するエントリを含む。

【００２６】

配列要素	配列要素内容
配列要素１	重要度０にて有効なサービス
配列要素２	重要度１にて有効なサービス
配列要素３	重要度２にて有効なサービス
配列要素４	重要度３にて有効なサービス
配列要素５	重要度４にて有効なサービス
配列要素６	重要度５にて有効なサービス
配列要素７	重要度６にて有効なサービス
配列要素８	未使用サービス

【００２７】サービス有効配列は、本願の出願人に権利譲渡された、１９９７年３月２８日付けのC. K. Eilertらによる米国特許出願第８２７５２９号"Managing Processor Resources in a Multisystem Environment"でも述べられている。

【００２８】作業負荷マネージャ１０５は、要求のサービス・クラスの重要度にて有効な最大限のサービスにより、システム１００上で新たなサーバを始動する。続くアドレス空間１６４は、作業負荷をサポートすることが要求されるときに始動される（後述のポリシ調整を参照）。好適には、アドレス空間１６４を始動する機構は、自動的にアドレス空間を始動する他の実施例における共通の問題を回避するための、幾つかのフィーチャを有する。従って、アドレス空間１６４の始動は好適には、１度に１つの始動だけが進行するように歩調を合わされる。この歩調合わせは、システム１００にアドレス空間１６４の始動が殺到することを回避する。

【００２９】また、所定の連続的な始動失敗（例えば３度の失敗）に遭遇するとき、所与のアプリケーション環境においては、追加のアドレス空間１６４の生成を回避するための特殊論理が、好適には提供される。こうした

失敗が起こり得る原因として、アプリケーション環境における、ＪＣＬプロシージャのＪＣＬエラーが考えられる。前述の特殊論理の提供は、ＪＣＬエラーが訂正されるまで、成功裡に始動しないアドレス空間を始動しようとするループに入り込むことを回避する。

【００３０】更に、作業要求１６２を実行する間に、サーバ・アドレス空間１６４が失敗する場合、作業負荷マネージャ１０５は好適には、それを置換するための新たなアドレス空間を始動する。失敗が繰り返されると、作業負荷マネージャ１０５は、オペレータ・コマンドにより問題が解決されたことを知らされるまで、そのアプリケーション環境において、作業要求の受諾を停止する。

【００３１】所与のサーバ・アドレス空間１６４は、たとえそれが通常、１つの作業キュー１６１だけをサービスするとしても、そのアプリケーション環境において、物理的にあらゆる作業要求１６２をサービスすることができる。好適には、サーバ・アドレス空間１６４は、もはやその作業キュー１６１をサポートする必要がないときでも、即時終了されない。代わりに、サーバ・アドレス空間１６４は、ある期間"フリー・エージェント"として待機し、同一のアプリケーション環境において、別の作業キュー１６１をサポートするために使用され得るか否かを調査する。サーバ・アドレス空間１６４が新たな作業キュー１６１にシフトされ得る場合、その作業キューに対して新たなサーバ・アドレス空間を始動するオーバヘッドが回避される。所定時間（例えば5分）内に、サーバ・アドレス空間１６４が別の作業キュー１６１により必要とされない場合、それは終了される。

【００３２】本発明は入力としてシステム管理者により確立され、データ記憶機構１４０に記憶された性能目標１４１及びサーバ定義を受け取る。データ記憶機構１４０は、管理される各システム１００によりアクセス可能である。ここで示される性能目標には２つのタイプ、すなわち応答時間（秒）と、実行速度（%）とがある。当業者であれば、本発明の趣旨及び範囲から逸れることなく、他の目標または追加の目標が選択され得ることが理解できよう。性能目標には、各目標の相対重要度の指定が含まれる。性能目標１４１が、管理される各システム１００のオペレーティング・システム１０１の作業負荷マネージャ要素１０５により、システム内に読出される。システム管理者により確立され、指定された性能目標の各々は、各システム１００上の作業負荷マネージャ１０５に、個々の作業単位が割当てられる性能クラスを確立させる。各性能クラスがクラス・テーブル・エントリ１０６により、オペレーティング・システム１０１のメモリ内に表される。（内部表現にて）指定された目標、及び性能クラスに関する他の情報が、クラス・テーブル・エントリに記録される。クラス・テーブル・エントリに記憶される他の情報には、サーバ１６３の数１０７（制御変数）、目標クラスの相対重要度１０８（入力値）、多重システム性能指標（ＰＩ）１５１、ローカル性能指標１５２（性能測定を表す計算値）、応答時間目標１１０（入力値）、実行速度目標１１１（入力値）、サンプル・データ１１３（測定データ）、リモート応答時間履歴（１５７）（測定データ）、リモート速度履歴１５８（測定データ）、サンプル・データ履歴１２５（測定データ）、及び応答時間履歴１２６（測定データ）が含まれる。

【００３３】オペレーティング・システム１０１はシステム資源マネージャ（ＳＲＭ）１１２を含み、これは多重システム目標駆動型制御装置（ＭＧＤＰＣ）１１４を含む。これらの構成要素は一般に、米国特許第５４７３７７３号及び同第５６７５７３９号で述べられるように動作する。しかしながら、ＭＧＤＰＣ１１４は本発明に従い、サーバ１６３の数を管理するように変更される。ＭＧＤＰＣ１１４は後述のように、目標の達成度を測定する機能、改善された性能を必要とするユーザ性能目標クラスを選択する機能、及び、関連作業単位の制御変数を変更することにより、選択されたユーザ性能目標クラスの性能を改善する機能を実行する。好適な実施例では、ＭＧＤＰＣ機能はおよそ毎１０秒ごとの周期的なタイマ満了にもとづき、周期的に実行される。ＭＧＤＰＣ機能が実行されるインタバルは、ＭＧＤＰＣインタバルまたはポリシ調整インタバルと呼ばれる。

【００３４】ＭＧＤＰＣ１１４の動作の一般的な態様は、米国特許第５６７５７３９号で述べられるように、次のようである。ブロック１１５で、各ユーザ性能目標クラス１０６に対して、指定目標１１０または１１１を用いて、多重システム性能指標１５１及びローカル性能指標１５２が計算される。多重システム性能指標１５１は、管理される全てのシステム１００に渡る目標クラスに関連付けられる作業単位の性能を表す。ローカル性能指標１５２は、ローカル・システム１００上の目標クラスに関連付けられる作業単位の性能を表す。結果の性能指標１５１、１５２は、対応するクラス・テーブル・エントリ１０６に記録される。ユーザ性能目標の達成度を測定する方法としての性能指標の概念は周知である。例えば、前記のFergusonらによる米国特許第５５０４８９４号では、性能指標として、実際の応答時間が目標応答時間により除算されるように述べられている。

【００３５】ブロック１１６で、ユーザ性能目標クラスが選択され、相対目標重要度１０８及び性能指標１５１、１５２の現在値の順で、性能改善度を受信する。選択されたユーザ性能目標クラスは、レシーバ（receiver）と呼ばれる。ＭＧＤＰＣ１１４はレシーバを選択するとき、最初に多重システム性能指標１５１を用いることにより、管理される全てのシステム１００に渡り作業単位が目標を満足するために、最も大きな影響を有するアクションを実行する。多重システム性能指標１５１にもとづき実行すべきアクションが存在しない場合、ロー

カル性能指標１５２が用いられ、ローカル・システム１００がその目標を満足するために最も有用なレシーバが選択される。

【００３６】候補のレシーバ・クラスが決定された後、周知のように、状態サンプル１２５を用いて性能ボトルネックを構成する、そのクラスの制御変数がブロック１１７で決定される。米国特許第５６７５７３９号で述べられるように、制御変数には、保護プロセッサ記憶ターゲット（ページング遅延に影響）、スワップ保護時間（ＳＰＴ）ターゲット（スワップ遅延に影響）、多重プログラミング・レベル（ＭＰＬ）・ターゲット（ＭＰＬ遅延に影響）、及びディスパッチ優先順位（ＣＰＵ遅延に影響）などの変数が含まれる。本発明によれば、制御変数として、キュー遅延に影響を及ぼすサーバ１６３の数が含まれる。

【００３７】図１では、サーバ１６３の数１０７が、クラス・テーブル・エントリ１０６に記憶されて示され、これは１クラス当たり、１つのキュー１６１が限度であることを意味する。しかしながら、これは単に説明の簡略化のためであり、当業者であれば、単にデータの位置を変更することにより、１クラス当たり複数のキュー１６１が独立に管理され得ることが理解できよう。基本的な必要条件は、１つのキュー１６１に対する作業要求１６２が１つの目標だけを有すること、各サーバ１６３が要求をサービスするための等しい能力を有すること、及び作業負荷マネージャ１０５からの（への）通知無しでは、サーバが２つ以上のキュー１６１上の作業をサービスすることができないことである。

【００３８】候補の性能ボトルネックが識別された後、制御変数の潜在的な変化がブロック１１８で考慮される。ブロック１２３で、相対目標重要度１０８及び性能指標１５１、１５２の現在値にもとづき、性能低下が起こり得るユーザ性能目標クラスが選択される。従って、選択されたユーザ性能目標クラスはドナーと呼ばれる。

【００３９】候補ドナー・クラスが選択された後、提案された変化がブロック１２４で評価される。具体的には、サーバ１６３の数１０７、及び前述され、また米国特許第５６７５７３９号でも述べられる変数を含む制御変数の各々に対して、レシーバ及びドナーの両者において、多重システム性能指標１５１及びローカル性能指標１５２の期待変化に対するネット値が評価される。提案された変化は、結果が目標に対して、ドナーに与える損害よりも多くの改善をレシーバにもたらす場合、ネット値を有する。提案された変化がネット値を有する場合、それぞれの制御変数がドナー及びレシーバの両者に対して調整される。

【００４０】管理される各システム１００はデータ伝送機構１５５に接続され、これは各システム１００がデータ・レコードをあらゆる他のシステム１００に送信することを可能にする。ブロック１５３で、各目標クラスの最近の性能を記述するデータ・レコードが、あらゆる他のシステム１００に送信される。

【００４１】多重システム目標駆動型性能制御装置（ＭＧＤＰＣ）機能は、周期的に実行され（好適な実施例では毎１０秒ごとに１度）、タイマ満了を介して呼び出される。ＭＧＤＰＣの機能は、性能問題の増分検出及び訂正のためのフィードバック・ループを提供し、オペレーティング・システム１０１を適応され、自己同調させる。

【００４２】米国特許第５６７５７３９号で述べられるように、ＭＧＤＰＣインタバルの終わりに、インタバルの間の各目標クラスの性能を記述するデータ・レコードが、管理される各リモート・システム１００に送信される。応答時間目標を有する性能目標クラスに対して、このデータ・レコードは目標クラス名と、リモート応答時間履歴の行（row）に等価なエントリを有する配列とを含む。ここでリモート応答時間履歴は、最後のＭＧＤＰＣインタバルに渡る目標クラスの完了を記述する。速度目標を有する目標クラスに対して、このデータ・レコードは目標クラス名と、目標クラス内の作業が最後のＭＧＤＰＣインタバル内で実行中にサンプリングされた回数と、目標クラス内の作業が最後のＭＧＤＰＣインタバル内で、実行中または遅延中にサンプリングされた回数とを含む。本発明によれば、各システム１００が追加のデータとして、データを送信するシステム１００のサービス有効配列、各キュー１６１に対するサーバ１６３の数、及び各キュー１６１に対する遊休サーバ１６３の数を送信する。

【００４３】ブロック１５４で、リモート・データ・レシーバがＭＧＤＰＣ１１４とは非同期に、リモート・システム１００から性能データを受信する。受信データはＭＧＤＰＣ１１４による後の処理のために、リモート性能データ履歴（１５７、１５８）に配置される。

【００４４】図３は、解決する資源ボトルネックを選択するために、ボトルネック発見手段１１７により使用される状態データを示す。各遅延タイプに対して、性能目標クラス・テーブル・エントリ１０６が、その遅延タイプに遭遇するサンプルの数、及びＭＧＤＰＣ１１４の現呼び出しの間に、その遅延タイプが既にボトルネックとして選択されたか否かを示すフラグを含む。クロス・メモリ・ページング・タイプ遅延の場合、クラス・テーブル・エントリ１０６が、遅延に遭遇したアドレス空間の識別子を含む。

【００４５】ボトルネック発見手段１１７の論理フローが図４に示される。解決するボトルネックの選択は、ＭＧＤＰＣ１１４の現呼び出しの間にまだ選択されておらず、最も多くのサンプルを有する遅延タイプを選択することにより実行される。遅延タイプが選択されると、フラグがセットされ、それによりボトルネック発見手段１１７が、ＭＧＤＰＣ１１４のこの呼び出しの間に再度呼

び出される場合、その遅延タイプがスキップされる。

【００４６】図４のステップ５０１では、ＣＰＵ遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ５０２で、ＣＰＵ遅延選択フラグがセットされ、ＣＰＵ遅延が次に解決されるべきボトルネックとして返却される。

【００４７】ステップ５０３では、ＭＰＬ遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ５０４で、ＭＰＬ遅延選択フラグがセットされ、ＭＰＬ遅延が次に解決されるべきボトルネックとして返却される。

【００４８】ステップ５０５では、スワップ遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ５０６で、スワップ遅延選択フラグがセットされ、スワップ遅延が次に解決されるべきボトルネックとして返却される。

【００４９】ステップ５０７では、ページング遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。肯定の場合、ステップ５０８で、ページング遅延選択フラグがセットされ、ページング遅延が次に解決されるべきボトルネックとして返却される。５つのタイプのページング遅延が存在する。ステップ５０７では、最も多くの遅延サンプルを有するタイプが突き止められ、ステップ５０８で、特定のタイプに対してフラグがセットされ、その特定のタイプが返却されれる。好適な実施例の環境（ＯＳ／３９０）において周知のように、ページング遅延のタイプには、専用領域、共通領域、クロス・メモリ、仮想入出力（ＶＩＯ）、及びハイパ空間が存在し、各々がページング遅延状況に対応する。

【００５０】最後にステップ５０９で、キュー遅延タイプが、まだ選択されていない全ての遅延タイプの中で、最も多くの遅延サンプルを有するか否かが判断される。クラスはキュー１６１上の各作業要求に対して、ローカル・システム１００上で実行される資格のある１つのキュー遅延タイプ・サンプルを獲得する。肯定の場合、ステップ５１０で、キュー遅延選択フラグがセットされ、キュー遅延が次に解決されるべきボトルネックとして返却される。キュー遅延は、クラスタ９０内の別のシステム１００が最後のポリシ調整インタバルの間に、キュー１６１に対してサーバ１６３を始動した場合、ローカル・システム１００上では解決されない。キュー遅延はまた、候補レシーバ・クラスが準備完了の作業をスワップ・アウトした場合にも、解決されない。

【００５１】次のセクションでは、ボトルネック発見手段１１７により選択された遅延を低減するように、制御変数を変更することにより、如何にレシーバ性能目標クラス性能が改善されるか、そして特に、レシーバにより遭遇されるキュー遅延を低減することにより、如何に性能が改善されるかについて述べる。共用キュー１６１の場合、これは２ステップ・プロセスである。第１に、ローカル・システム１００上にサーバ１６３を追加することにより、評価が行われる。この評価には、ドナー作業に対する影響が含まれる。サーバ１６３の追加において、ネット値が存在する場合、次のステップでサーバがローカル・システム１００上で始動されるべきか、或いはそれらがクラスタ９０内の別のシステム１００上で始動されるべきかが決定される。リモート・システム１００がサーバ１６３を始動するために、より好適であると思われる場合、ローカル・システム１００はそのリモート・システムにサーバを始動する機会を与えるために待機する。しかしながら、そのリモート・システム１００がサーバ１６３を始動しない場合、ローカル・システム１００が図９に関連して後述されるように、それらを始動する。

【００５２】図５は、追加のサーバ１６３を始動することによる、性能の改善を評価する論理フローを示す。図５乃至図７は、固定手段１１８によりネット値手段１２４に提供される性能指標デルタ予測を生成するステップを示す。ステップ１４０１で、サーバ１６３の新たな数が評価のために選択される。この数は変化を価値あるものにするために、十分なレシーバ値（ステップ１４０５でチェックされる）を生成するように十分大きくなければならない。一方、この数は、追加のサーバ１６３の値が限られるように、例えばキューに待機されて実行される作業要求の総数より大きくならないように、余り大きくてはならない。次のステップは、追加のサーバ１６３が使用する追加のＣＰＵを計算する。これは作業要求により使用される平均ＣＰＵに、追加されるサーバ１６３を乗算することにより実行される。

【００５３】ステップ１４０２で、新たな数のサーバ１６３における作業要求１６２の予測数が、図６に示されるサーバ準備完了ユーザ平均グラフから読出される。ステップ１４０３で、現予測キュー遅延が、図７に示されるキュー遅延グラフから読出される。ステップ１４０４で、ローカル性能指標デルタ及び多重システム性能指標デルタの予測値が計算される。これらの計算は以下で示される。

【００５４】図６は、キュー準備完了ユーザ平均グラフを示す。キュー準備完了ユーザ平均グラフは、キュー１６１に対するサーバ１６３の数の変化を評価するときに、サーバ１６３に対する需要を予測するために使用される。このグラフは、作業要求１６２がバックアップを開始するポイントを示す。横座標（ｘ）値は、キュー１６１が使用可能なサーバ１６３の数である。縦座標（ｙ）値は、実行準備完了の作業要求１６２の最大数である。

【００５５】図７は、キュー遅延グラフを表す。キュー遅延グラフは、キュー１６１に対するサーバ１６３の数を増減する値を評価するために使用される。このグラフは、キュー・サーバ１６３の数を増加することにより、如何に応答時間が改善されるか、或いはキュー・サーバ１６３の数を減少することにより、如何に応答時間が低下されるかを示す。グラフはまた、例えばデータベース・ロック競合などの、追加のサーバ１６３を追加することにより生じ得る、作業負荷マネージャ１０５により管理されない資源に対する競合を暗黙的に考慮する。こうした場合では、追加のサーバ１６３が追加されても、グラフ上のキュー遅延が減少しない。横座標値は、使用可能なサーバ１６３を有し、システム１００のクラスタ９０に渡りスワップ・インされる準備完了の作業要求１６２の割合である。縦座標値は１完了当たりのキュー遅延である。

【００５６】サーバ１６３の数の増加に対する、シスプレックス（すなわち多重システム）性能指標（ＰＩ）デルタは、次のように計算される。ここでシスプレックス性能指標デルタが計算される理由は、キュー１６１がシスプレックス全体に渡る資源であるからである。

【００５７】応答時間目標に対して、

（予測シスプレックスＰＩデルタ）＝（予測キュー遅延－現キュー遅延）／応答時間目標

【００５８】速度目標に対して、

（新シスプレックス速度）＝｛cpuu＋（（cpuu／oldserver）*newserver）｝／｛non-idle＋（（qd/qreq）*（oldserver－newserver））｝

（シスプレックスＰＩデルタ）＝（現シスプレックスＰＩ－目標）／新シスプレックス速度

【００５９】ここでcpuuは、シスプレックスＣＰＵ使用サンプル；oldserverは、評価される変化が実行される前のサーバ１６３の数；newserverは、評価される変化が実行された後のサーバ１６３の数；non-idleは、シスプレックス非遊休サンプルの総数；qdは、シスプレックス・キュー遅延サンプル；qreqは、キュー１６１上の作業要求１６２の数である。

【００６０】同様の計算が、サーバ１６３の数の減少に対する性能指標デルタを計算するために使用される。

【００６１】ステップ１４０５で、追加のサーバ１６３の数により提供される十分なレシーバ値に対してチェックが行われる。好適には、このステップは、新たなサーバ１６３が、それらの追加を価値あるものにするために十分なＣＰＵ時間を獲得するか否かを判断するステップを含む。十分なレシーバ値が存在しない場合、制御はステップ１４０１に戻り、より多くのサーバ１６３が評価のために選択される。

【００６２】十分なレシーバ値が存在する場合、ステップ１４０６でドナー選択手段１２３が呼び出され、レシーバ性能目標クラスのために、追加のサーバ１６３を始動するために必要な記憶装置のドナーを見い出す。

【００６３】ドナー・クラスのために調整される制御変数は、必ずしもそのクラスに対するサーバ１６３の数１０７である必要はない。ドナー・クラスの幾つかの異なる制御変数の任意の１つ、例えばＭＰＬスロットまたは保護プロセッサ記憶装置などが、代わりにまたは追加として調整され、追加のサーバ１６３のために必要な記憶装置を提供してもよい。本発明の一部を成すものではないが、こうした制御変数の調整がドナー・クラスに与える効果を評価する方法が、米国特許第５５３７５４２号及び同第５６７５７３９号で述べられている。

【００６４】ステップ１４０７では、ドナーから記憶装置を受け取り、レシーバ・クラスのためにサーバ１６３の数を増加するに際して、ネット値が存在することを保証するためにチェックが行われる。米国特許第５６７５７３９号で述べられるように、これは１つ以上の幾つかの異なる基準、例えば資源再割当ての後に、ドナーがその目標に合致するように予測されるか否か、レシーバが現在その目標を逸しつつあるか否か、レシーバがドナーよりも重要なクラスか否か、またはドナー及びレシーバの結合性能指標において、ネット利得が存在するか否か、すなわち、サーバをレシーバ・クラスに追加することが、レシーバ・クラスの性能指標に及ぼすプラスの効果が、ドナー・クラスの性能指標に及ぼすマイナスの効果を上回るか否かなどの基準を用いて決定される。ネット値が存在する場合、次のステップで、ローカル・システム１００が新たなサーバ１６３を始動するための（ステップ１４０８）、クラスタ９０内の最善のシステムか否かが決定される。ネット値が存在しない場合、レシーバ目標クラス・キュー遅延問題は解決され得ない（ステップ１４０９）。

【００６５】図９は、新たなサーバ１６３を始動するクラスタ９０内の最善のシステム１００を表す、ターゲット・システムを決定するプロシージャを示す。このプロシージャは、一旦サーバの追加に対してネット値が存在し、１つ以上のサーバ１６３がレシーバ・クラスに追加されるべきことが決定されると、図５のステップ１４０８の一部として、クラスタ９０内の各システム１００により実行される。ローカル・システム１００が最初に、クラスタ９０内の任意のシステム１００が他の作業に影響すること無く、新たなサーバ１６３をサポートするための十分な遊休容量を有するか否かをチェックする（ステップ８０１）。これはクラスタ９０内の各システム１００のサービス有効配列を調査し、配列要素８において、新たなサーバ１６３をサポートするために使用可能な十分なＣＰＵサービス（未使用ＣＰＵサービス）を有するシステム１００を選択することにより実行される。複数のシステム１００が十分な未使用ＣＰＵサービスを有する場合、最も多くの未使用サービスを有するシステム１００が選択される。しかしながら、キューに待機さ

れた作業要求が存在するときに、システム１００が遊休状態のサーバを有する場合、これは多くの作業要求がそのシステム１００に対して親和性を持たないことを意味するので、この遊休状態のサーバ１６３を有するシステム１００は選択されない。

【００６６】ローカル・システム１００は次に、サーバ１６３を始動するための、十分な遊休状態のＣＰＵ容量を有するターゲット・システム１００が見い出されたか否かをチェックする（ステップ８０２）。ターゲット・システム１００が見い出され、それがローカル・システム１００の場合（ステップ８０３）、ローカル・システム１００がサーバ１６３を局所的に始動する（ステップ８０４乃至８０５）。

【００６７】ステップ８０３で、別のシステム１００が新たなサーバ１６３を始動するための最も多くの遊休容量を有することが見い出されると、制御はステップ８０６に移行し、ローカル・システム１００がポリシ調整インタバル（１０秒）を待機し、別のシステム１００がサーバ１６３を始動したか否かをチェックする（ステップ８０７）。別のシステム１００がサーバ１６３を始動した場合、局所的に何もアクションは実行されない（ステップ８１１）。他のシステム１００がサーバ１６３を始動しなかった場合、ローカル・システム１００は、自身が新たなサーバ１６３をサポートする十分な遊休ＣＰＵ容量を有するか否かをチェックする（ステップ８１２）。有する場合、ローカル・システム１００は局所的にサーバ１６３を始動する（ステップ８１３乃至８１４）。

【００６８】新たなサーバ１６３をサポートするための十分な遊休ＣＰＵ容量を有するシステム１００が存在しなかった場合、或いは、十分な遊休ＣＰＵ容量を有するシステム１００が存在したが、そのシステム１００がサーバを始動しなかった場合、制御はステップ８０８に移行する。こうしたシステム１００がサーバ１６３を始動しない１つの理由は、メモリが欠乏するためである。この時点で、ドナー作業に影響を及ぼすこと無しに、新たなサーバ１６３が始動され得ないことが判明する。従って、ローカル・システム１００は、局所的なサーバ１６３の始動により、ドナー作業がその目標を逸するか否かをチェックする。ドナー作業がその目標を逸しない場合、ローカル・システム１００はサーバ１６３を局所的に始動する（ステップ８１７乃至８１８）。サーバ１６３の局所的な始動により、ドナー・クラスがその目標を逸する場合には、ローカル・システム１００はドナー作業への影響が最も小さいシステム１００を見い出す（ステップ８０９）。

【００６９】図１０は、図９のステップ８０９で、ドナー作業への影響が最も小さいシステム１００を決定するルーチンを示す。ルーチンは最初に、ドナー・クラスの名前及びドナーの性能指標（ＰＩ）デルタを、クラスタ９０内の他のシステムに送信する（ステップ９０１）。このドナー情報を交換することにより、レシーバ・クラスに対するサーバ１６３の追加を評価している各システム１００は、他の全てのシステム１００へのサーバ追加の影響を調査する。ルーチンは次に、他のシステム１００がそれらのドナー情報を送信することを可能にするために、１ポリシ・インタバル（示される実施例では１０秒）を待機する（ステップ９０２）。ルーチンは次に、ドナー・クラスが最も低い重要度を有するシステム１００を選択し（ステップ９０３）、選択されたシステム１００を呼び出しルーチンに返却して、図９のステップ８０９を完了する（ステップ９０５）。ドナー重要度に対等なものが存在する場合（ステップ９０４）、ルーチンはドナーの性能指標（ＰＩ）デルタが最小のシステム１００を選択し（ステップ９０６）、このシステム１００を呼び出しルーチンに返却する（ステップ９０７）。

【００７０】再度図９を参照して、ステップ８０９を完了後、ローカル・システム１００は、ドナーに対して最も小さな影響を有するとして選択されたシステム１００が、ローカル・システムであるか否かをチェックする（ステップ８１０）。そうである場合、ローカル・システム１００がサーバ１６３を局所的に始動する（ステップ８１７乃至８１８）。それ以外では、ローカル・システム１００は、別のシステム１００がサーバ１６３を始動することを可能にするために、１ポリシ・インタバル待機し（ステップ８１５）、このインタバルの終りに別のシステム１００がサーバ１６３を始動したか否かをチェックする（ステップ８１６）。別のシステム１００がサーバ１６３を始動した場合、ローカル・システム１００は何もアクションを起こさない（ステップ８１８）。他のシステム１００がサーバ１６３を始動しなかった場合、ローカル・システム１００はそれらを局所的に始動する（ステップ８１７乃至８１８）。

【００７１】図５のステップ１４０８では、特定の環境下のキュー１６１に対して、新たなサーバ１６３を始動する要求を一時的に延期する論理が含まれる。既存の作業への不要な影響を回避するために、新たなサーバ１６３を始動する並列要求が制限される。この歩調合わせは、オペレーティング・システム１０１に、追加のサーバ１６３を始動するための多くの並列要求が殺到し、混乱することを回避する。システム管理者により提供される、データ・レポジトリ１４１内の誤った情報の検出も実行され、新たなサーバ１６３が成功裡に始動され得ない程、サーバ定義情報が不正確な場合に、無限の再試行ループを阻止する。一旦サーバ１６３が始動されると、サーバが予期せず故障した場合に、それを自動的に置換する論理も含まれる。同一のサーバ定義情報を有するが、同一の作業マネージャ１６０に対して異なるキュー１６１をサービスする遊休サーバ１６３が、キュー間で移動され得ることにより、特定のキューに対してサーバ

１６３の数を増加する要求が満足され、完全に新たなサーバを始動するオーバヘッドを回避する。

【００７２】本発明は好適には、１つ以上のハードウェア・マシン上で実行されるソフトウェア（すなわち、プログラム記憶装置上で実現される命令のマシン読出し可能プログラム）として実現される。これまで特定の実施例について述べてきたが、当業者であれば、ここで特定的に述べられた以外の他の実施例も、本発明の趣旨から逸れることなく実現され得ることが明らかであろう。また、当業者であれば、様々な等価な要素が、ここで特定的に開示された要素の代わりに代用され得ることが理解されよう。同様に、ここで開示された実施例の変更及び組み合わせも明らかであろう。例えば、各サービス・クラスに対して、ここで開示された１つのキューではなく、複数のキューが提供され得る。開示された実施例及びそれらの詳細は、本発明の実施例を教示することを意図したのものであって、本発明を制限するものではない。従って、こうした明白ではあるが、ここで開示されなかった変更及び組み合わせも、本発明の趣旨及び範囲に含まれるものと見なされる。

【００７３】まとめとして、本発明の構成に関して以下の事項を開示する。

【００７４】（１）情報処理システムのクラスタ内においてサーバの数を制御する方法であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の１つ以上のサーバにより処理されるものにおいて、１つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定するステップと、１つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップと、前記サーバを前記ターゲット・システム上に追加するステップとを含む、方法。

（２）前記決定するステップが前記システムの各々により実行される、前記（１）記載の方法。

（３）前記システムの各々が、該システムがターゲット・システムか否かを決定し、前記ターゲット・システムであると決定するとき、１つ以上のサーバを局所的に追加する、前記（２）記載の方法。

（４）前記サービス・クラスが第１のサービス・クラスであり、前記クラスタが少なくとも１つの他のサービス・クラスを有し、前記サービス・クラスの各々が、それぞれに対して定義された性能測定を有する、前記（１）記載の方法。

（５）サーバが前記第１のサービス・クラスに追加されるべきか否かを決定するステップが、所定数のサーバを前記第１のサービス・クラスに追加することが、前記第１のサービス・クラスの前記性能測定に及ぼすプラスの効果を決定するステップと、前記所定数のサーバを前記第１のサービス・クラスに追加することが、１つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を決定するステップと、前記第１のサービス・クラスの前記性能測定に及ぼすプラスの効果が、１つ以上の他のサービス・クラスの前記性能測定に及ぼすマイナスの効果を上回るか否かを決定するステップとを含む、前記（４）記載の方法。

（６）サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、前記クラスタ内の任意のシステムが、１つ以上の追加のサーバを追加するための十分な遊休容量を有するか否かを決定するステップと、前記クラスタ内の任意のシステムが、１つ以上の追加のサーバを追加するための十分な遊休容量を有する場合、１つのこうしたシステムを前記ターゲット・システムとして選択するステップとを含む、前記（１）記載の方法。

（７）最大の未使用容量を有するシステムが前記ターゲット・システムとして選択される、前記（６）記載の方法。

（８）サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定するステップが、前記クラスタ内のいずれのシステムも、１つ以上の追加のサーバを追加するための十分な遊休容量を有さない場合、１つ以上のサーバの追加が他の作業に最小の影響を及ぼす前記クラスタ内のシステムを決定するステップと、そのシステムを前記ターゲット・システムとして選択するステップとを含む、前記（６）記載の方法。

（９）マシンにより読出され、前記マシンにより実行されて、前記（１）記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。

（１０）情報処理システムのクラスタ内においてサーバの数を制御する装置であって、あるサービス・クラスに属する入来作業要求が、前記クラスタ全体に渡るキューに配置され、前記クラスタの前記システム上の１つ以上のサーバにより処理されるものにおいて、１つ以上のサーバが前記サービス・クラスに追加されるべきか否かを決定する手段と、１つ以上のサーバが前記サービス・クラスに追加されるべきと決定される場合、前記サーバが追加されるべき、前記クラスタ内のターゲット・システムを決定する手段と、前記サーバを前記ターゲット・システム上に追加する手段とを含む、装置。

（１１）情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する方法であって、入来作業要求が前記キューに配置され、前記システム上の１つ以上のサーバにより処理されるものにおいて、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップと、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記

キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動するステップとを含む、方法。

（１２）前記決定するステップが周期的インタバルで実行される、前記（１１）記載の方法。

（１３）異なるサービス・クラスに割当てられる作業要求が、異なるキューに配置され、前記決定するステップが前記キューの各々に対して実行される、前記（１１）記載の方法。

（１４）前記決定するステップが、前記クラスタ内の各システムにより実行される、前記（１１）記載の方法。

（１５）前記決定するステップが、前記キューが前記システム上にサーバを有するか否かを決定するステップと、前記キューが前記システム上にサーバを有さない場合、前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定するステップとを含む、前記（１４）記載の方法。

（１６）マシンにより読出され、前記マシンにより実行されて、前記（１１）記載の方法を実行する命令のプログラムを実現する、プログラム記憶装置。

（１７）情報処理システムのクラスタ内において、キュー内の各作業要求を処理可能なサーバの可用性を保証する装置であって、入来作業要求が前記キューに配置され、前記システム上の１つ以上のサーバにより処理されるものにおいて、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在するか否かを決定する手段と、前記キューに対してサーバを有さない前記クラスタのサブセットだけに親和性を有する作業要求が、前記キュー内に存在すると決定された場合、前記作業要求が親和性を有する前記サブセット内のシステム上で、前記キューに対してサーバを始動する手段とを含む、装置。

【図面の簡単な説明】

【図１】本発明のために適応化された制御オペレーティング・システム及びシステム資源マネージャ要素を有する、コンピュータ・システムを示すシステム構造図である。

【図２】ネットワークから、本発明の作業負荷マネージャにより管理されるサーバ・アドレス空間へのクライアント作業要求の流れを示す図である。

【図３】資源ボトルネックを選択するために使用される状態データを示す図である。

【図４】ボトルネック発見機能の論理フローを示すフローチャートである。

【図５】サーバの数を増加することにより、改善された性能をアクセスするステップのフローチャートである。

【図６】キュー準備完了ユーザ平均のサンプル・グラフである。

【図７】キュー遅延のサンプル・グラフである。

【図８】クラスタ内のどこかに、キュー上の各要求を実行可能な少なくとも１つのサーバが存在することを保証するプロシージャを示す図である。

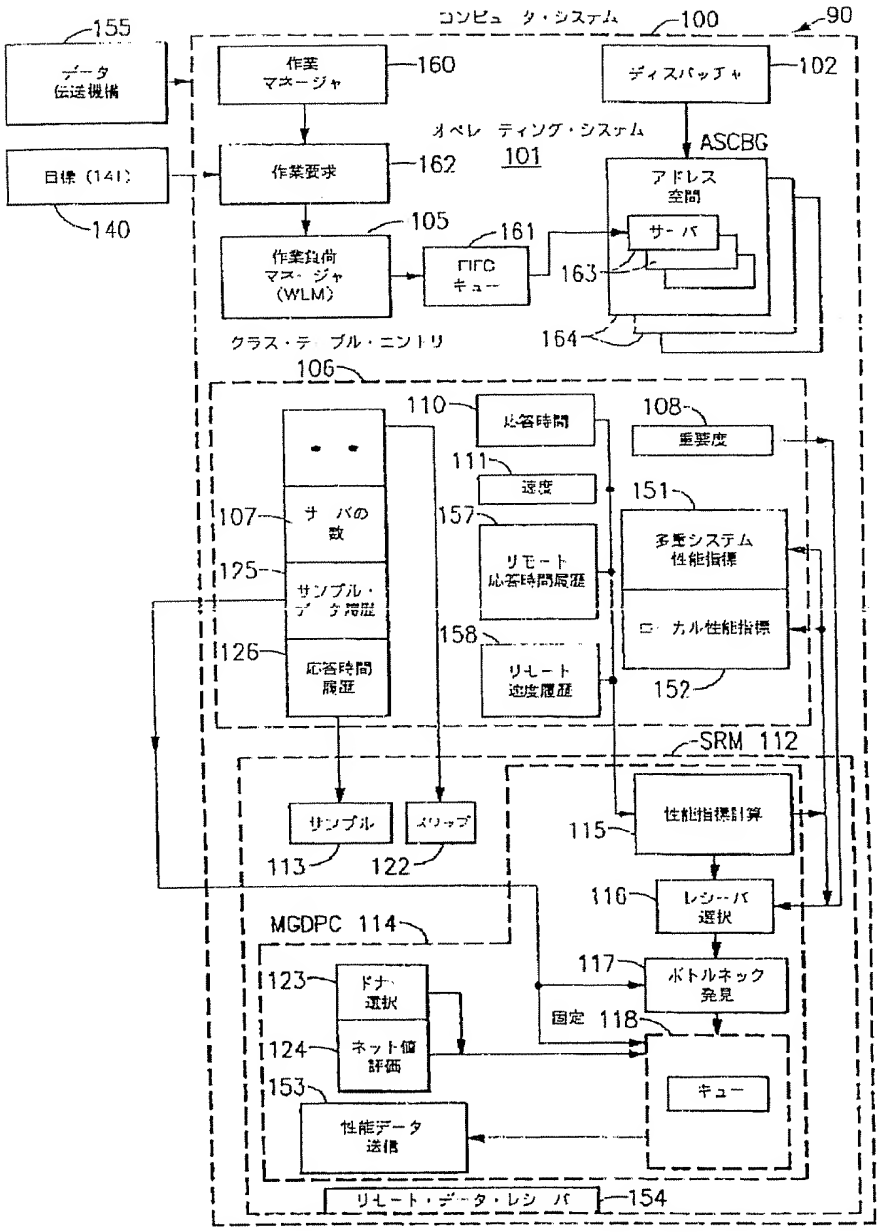
【図９】サーバを始動すべきクラスタ内の最善のシステムを決定するプロシージャを示す図である。

【図１０】ドナーへの影響が最も小さいシステムを見い出すプロシージャを示す図である。

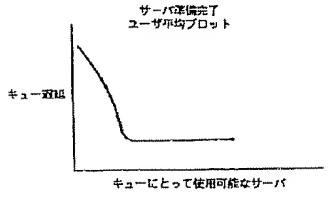
【符号の説明】

９０　クラスタ
１００　コンピュータ・システム
１０１　オペレーティング・システム
１０２　ディスパッチャ
１０５　作業負荷マネージャ（ＷＬＭ）
１０６　クラス・テーブル・エントリ
１０７　サーバの数
１１０　応答時間目標
１１１　実行速度目標
１１２　システム資源マネージャ（ＳＲＭ）
１１３　サンプル・データ
１１４　多重システム目標駆動型制御装置（ＭＧＤＰＣ）
１１７　ボトルネック発見手段
１１８　固定手段
１２３　ドナー選択手段
１２４　ネット値手段
１２５　サンプル・データ履歴
１２６　応答時間履歴
１４１　性能目標
１５１　多重システム性能指標
１５２　ローカル性能指標
１５５　データ伝送機構
１５７　リモート応答時間履歴
１５８　リモート速度履歴
１６０　作業マネージャ
１６１　キュー
１６２　作業要求
１６３　サーバ
１６４　アドレス空間

【図１】



【図６】

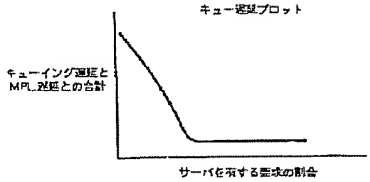


【図３】

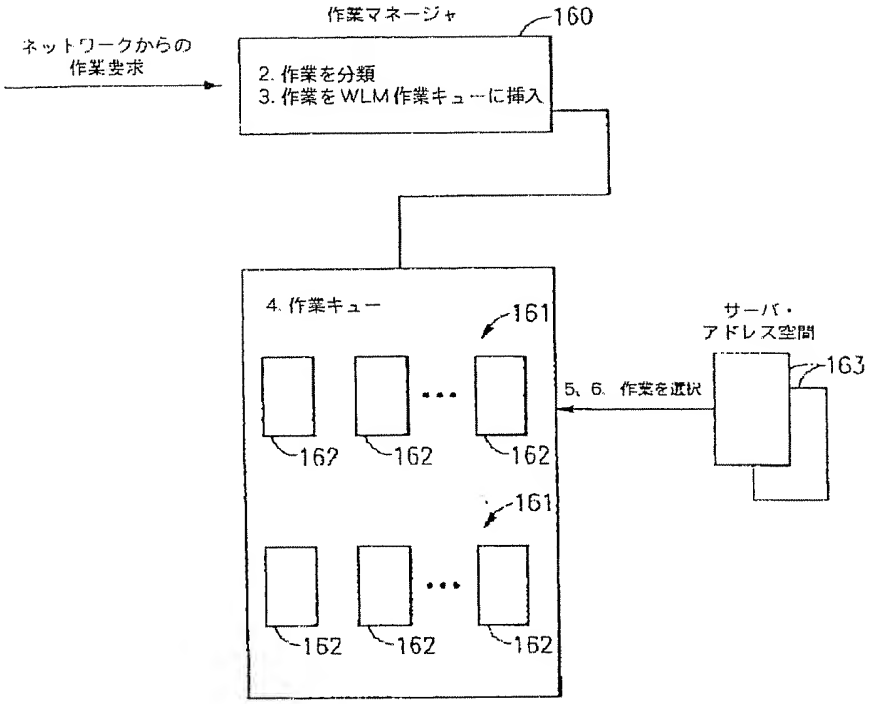
ボトルネック発見のための状態サンプル

CPU 遅延 サンプル	MPL 遅延 サンプル	スワップ 遅延 サンプル	AUX ページング 遅延 サンプル	・・・	キュー 遅延 サンプル
フラグ	フラグ	フラグ	フラグ	・・・	フラグ

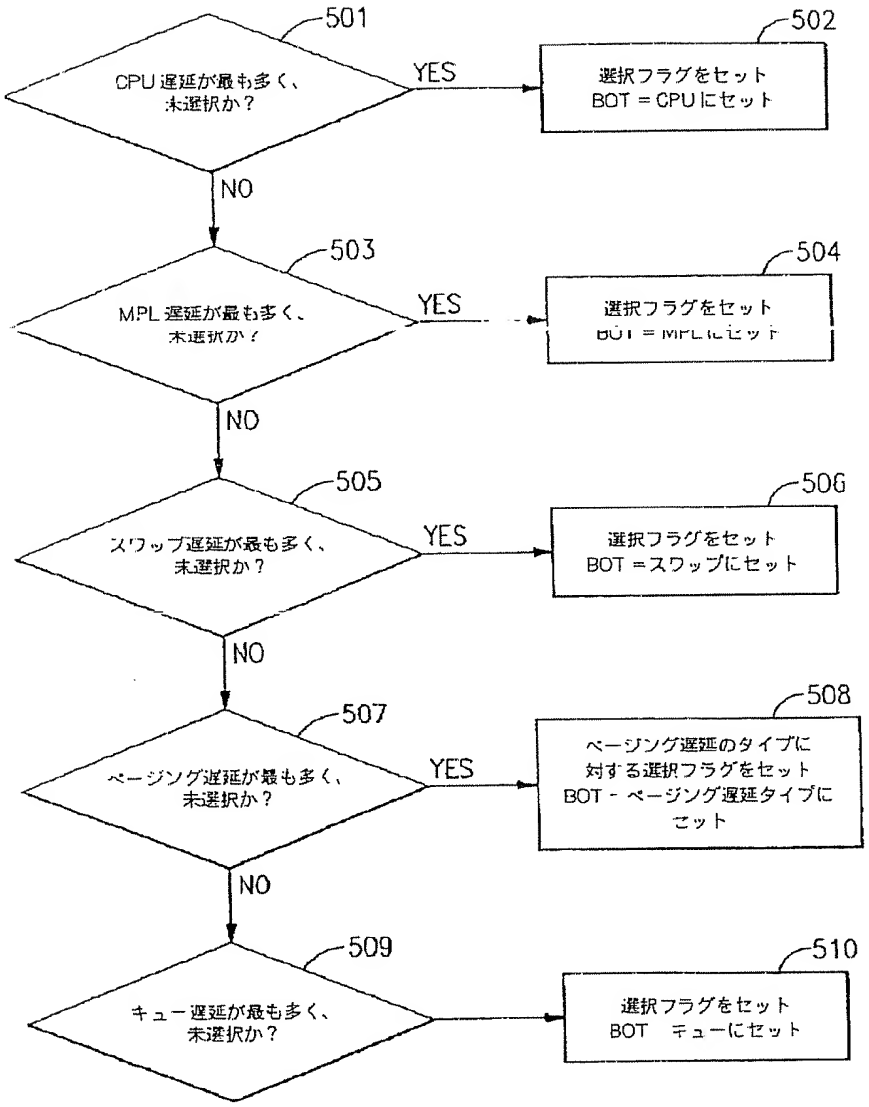
【図７】



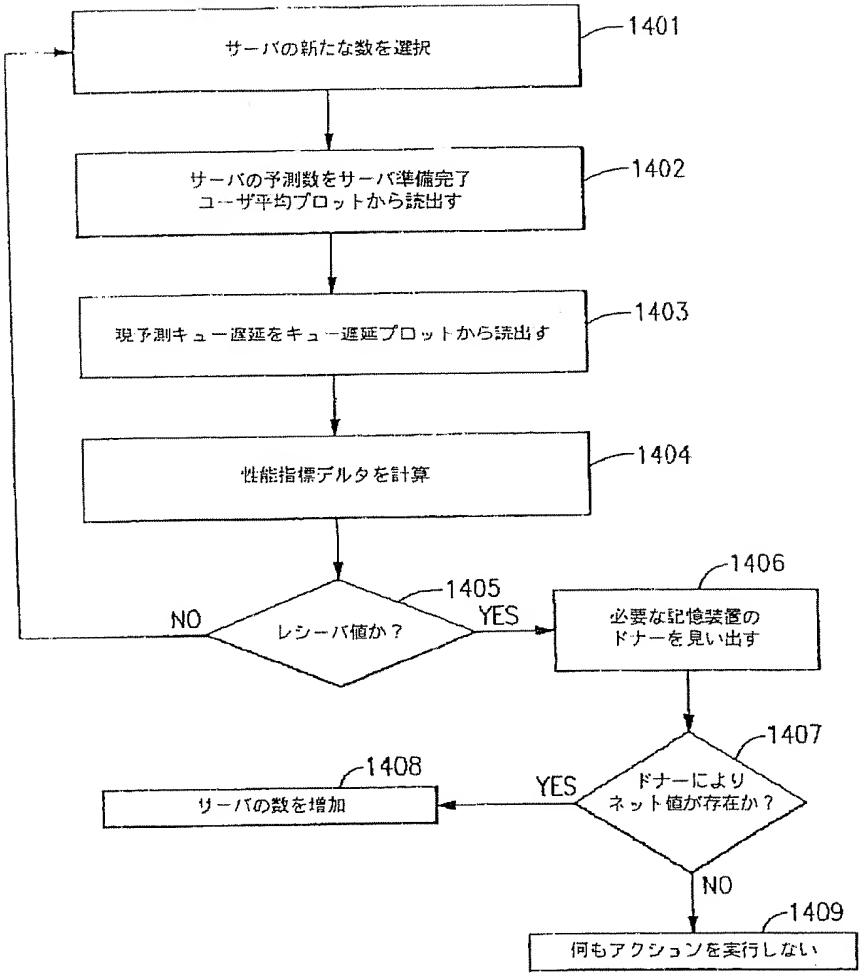
【図２】


作業マネージャ
160
ネットワークからの
作業要求
2. 作業を分類
3. 作業をWLM作業キューに挿入
4. 作業キュー
161
162
162
162
161
162
162
162
サーバ・
アドレス空間
163
5、6. 作業を選択

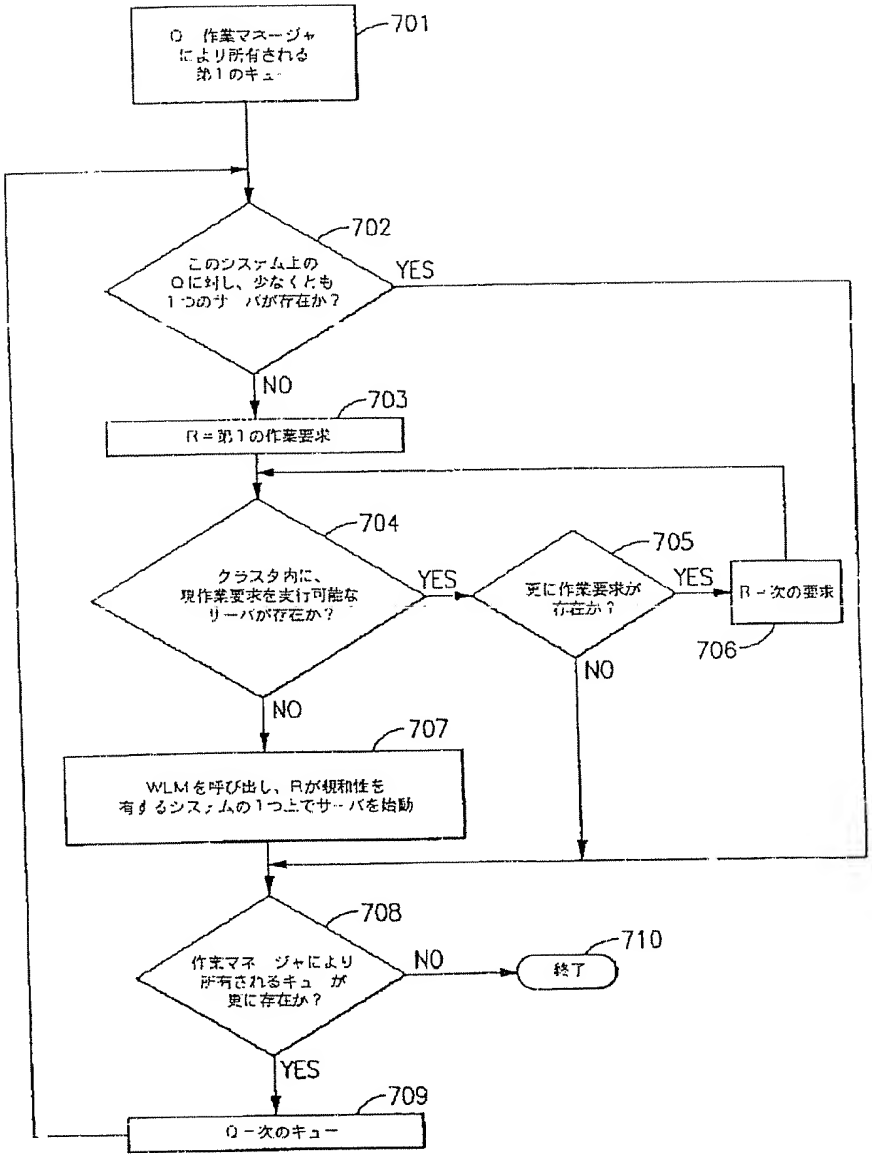
【図４】


501
CPU遅延が最も多く、未選択か？
YES
502
選択フラグをセット
BOT＝CPUにセット
NO
503
MPL遅延が最も多く、未選択か？
YES
504
選択フラグをセット
BOT＝MPLにセット
NO
505
スワップ遅延が最も多く、未選択か？
YES
506
選択フラグをセット
BOT＝スワップにセット
NO
507
ページング遅延が最も多く、未選択か？
YES
508
ページング遅延のタイプに対する選択フラグをセット
BOT－ページング遅延タイプにセット
NO
509
キュー遅延が最も多く、未選択か？
510
選択フラグをセット
BOT　キューにセット

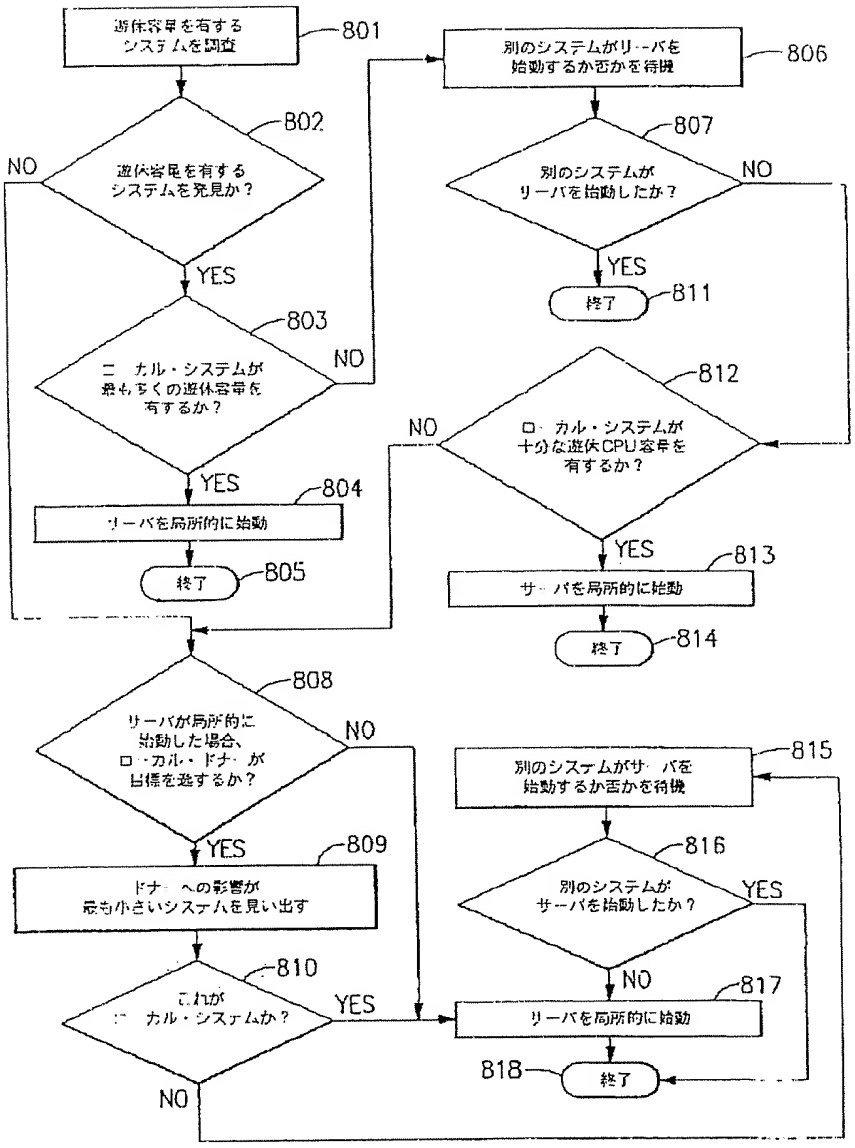
【図５】


サーバの新たな数を選択
1401
サーバの予測数をサーバ準備完了
ユーザ平均プロットから読出す
1402
現予測キュー遅延をキュー遅延プロットから読出す
1403
性能指標デルタを計算
1404
1405
NO
レシーバ値か？
YES
1406
必要な記憶装置の
ドナーを見い出す
1407
ドナーにより
ネット値が存在か？
YES
1408
サーバの数を増加
NO
1409
何もアクションを実行しない

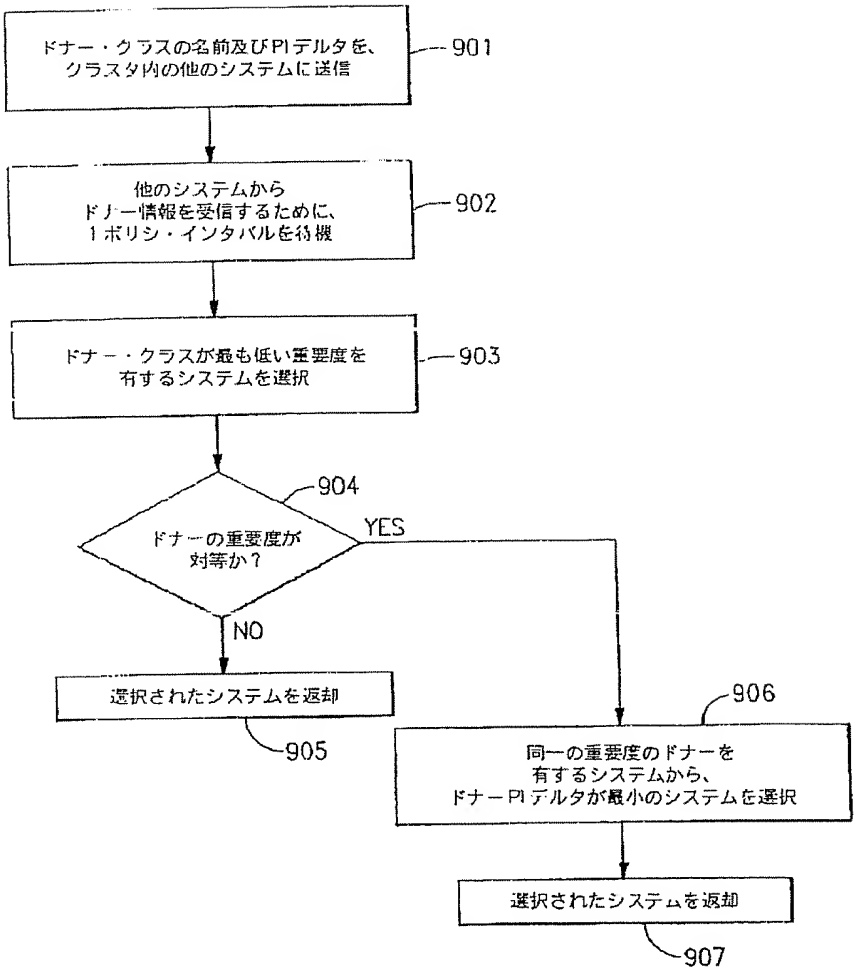
【図８】


Q　作業マネージャ
により所有される
第１のキュー
701
702
このシステム上の
Qに対し、少なくとも
１つのサーバが存在か？
YES
NO
703
R＝第１の作業要求
704
クラスタ内に、
現作業要求を実行可能な
リーバが存在か？
YES
705
更に作業要求が
存在か？
YES
R－次の要求
706
NO
NO
707
WLMを呼び出し、Rが規和性を
有するシステムの１つ上でサーバを始動
708
作業マネージャにより
所有されるキュー　が
更に存在か？
NO
710
終了
YES
709
Q－次のキュー

【図９】


遊休容量を有する
システムを調査
801
別のシステムがリーバを
始動するか否かを待機
806
遊休容量を有する
システムを発見か？
802
NO
YES
別のシステムが
リーバを始動したか？
807
NO
YES
終了
811
ニ　カル・システムが
最も多くの遊休容量を
有するか？
803
NO
YES
ロ・カル・システムが
十分な遊休CPU容量を
有するか？
812
NO
YES
リーバを局所的に始動
804
終了
805
サ・バを局所的に始動
813
終了
814
リーバが局所的に
始動した場合、
ロ・カル・ドナ・が
目標を遂するか？
808
NO
YES
別のシステムがサ・バを
始動するか否かを待機
815
ドナ・への影響が
最も小さいシステムを見い出す
809
別のシステムが
サーバを始動したか？
816
YES
NO
これが
ニ　カル・システムか？
810
YES
NO
リーバを局所的に始動
817
終了
818

【図１０】



フロントページの続き

(72)発明者 キャサリン・ケイ・エイラート
アメリカ合衆国12590、ニューヨーク州ワッピンガーズ・フォールズ、シャーウッド・ハイツ・ドライブ 34

(72)発明者 ジョン・イー・アーウィ
アメリカ合衆国12603、ニューヨーク州ポキプシ、グレンウッド・アベニュー 43